Primjena kartografskih sustava na Android platformi

Haag, Nikolas

SVEUČILIŠTE U RIJECI
ODJEL ZA INFORMATIKU

Nikolas Haag

PRIMJENA KARTOGRAFSKIH SUSTAVA NA ANDROID PLATFORMI

ZAVRŠNI RAD

Rijeka, 2014.

SVEUČILIŠTE U RIJECI
ODJEL ZA INFORMATIKU

Nikolas Haag
Broj indeksa: 17508
Smjer: Dvopredmetna informatika
Prediplomski studij

PRIMJENA KARTOGRAFSKIH SUSTAVA NA ANDROID PLATFORMI

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Marina Ivašić-Kos

Rijeka, rujan 2014.

Sadržaj

1. Uvod

Android je slobodan operativni sustav otvorenog koda, kojeg razvijaju Google i Open Handset Alliance[1] specifično za mobilne uređaje. Cilj održavanja otvorene platforme je zajedničko prilagođavanje i poboljšavanje sustav od strane različitih razvojnih programera umjesto samo jedne centralizirane kompanije.

Mobilni telefoni sa sve boljom konfiguracijom otvorili su ogromno tržište mobilnih aplikacija, koje pogotovo radi ugrađenih senzora (koja na uobičajenim stolnim računalima nisu prisutna), čine odličnu bazu za razvoj raznih implementacija kartografskih sustava poput Google Maps ili Mapsforge[2].

Cilj ovog rada je prikazati mogućnosti implementacije dostupnih kartografskih sustava u razvoju android aplikacija. Istražit će se načini realizacije osnovnih navigacijskih aplikacija od odabira kartografskih sustava pa sve do gotove aplikacije.

U prvom poglavlju nakon uvoda opisuje se Android operativni sustav, te se navode relevantni podaci koji su bili presudni za odabir ove mobilne platforme. Slijedi poglavlje o razvojnoj okolini za programiranje aplikacija. Opisano je koja razvojna okolina se koristi, te koji dodatni razvojni alati su potrebni za prevođenje u izvršnu android aplikaciju i za testiranje i debugiranje iste. Za pokretanje koda, možemo na Google developers preuzeti razvojne alate.

U okviru rada potrebno je definirati neke pojmove geografije i geoinformatike kako bi terminologija bila jednoznačna, te se u četvrtom poglavlju objašnjavaju pojmovi i na koji način će se koristiti u okviru rada. Najvažniji pojam vjerojatno jesu geografske koordinate, koje služe definiranju pozicije

[1] Open Handset Alliance je udruženje ustanova koje se sastoji od 84 kompanija (od mobilnih operatera do proizvođača mobitela). Bavi se razvoj em otvorenih standarda za mobilne uređaje.
[2] Mapsforge je projekt koji se bavi razvojem otvorenog koda za rad sa OpenStreetMaps kartama na Android platformi. Trenutno se nalazi u verziji 0.4.3.

na zemljinoj površini na jednostavan i jednoznačan način. Koordinate se u tom slučaju sastoje od dvaju brojeva, geografske širine i geografske dužine.

Četvrto se poglavlje bavi vrstama kartografskih sustava. Uspoređuju se razni tržišni standardi, kako bi se odabrao najbolji za rješavanje zadatka. Ovo poglavlje služi kao uvod u peto poglavlje, gdje se razrađuju primjeri aplikacija koje primjenjuju razne biblioteke. Relevantni dijelovi koda su umetnuti i obrazloženi, dok su konkretni rezultati prikazani slikama ekrana.

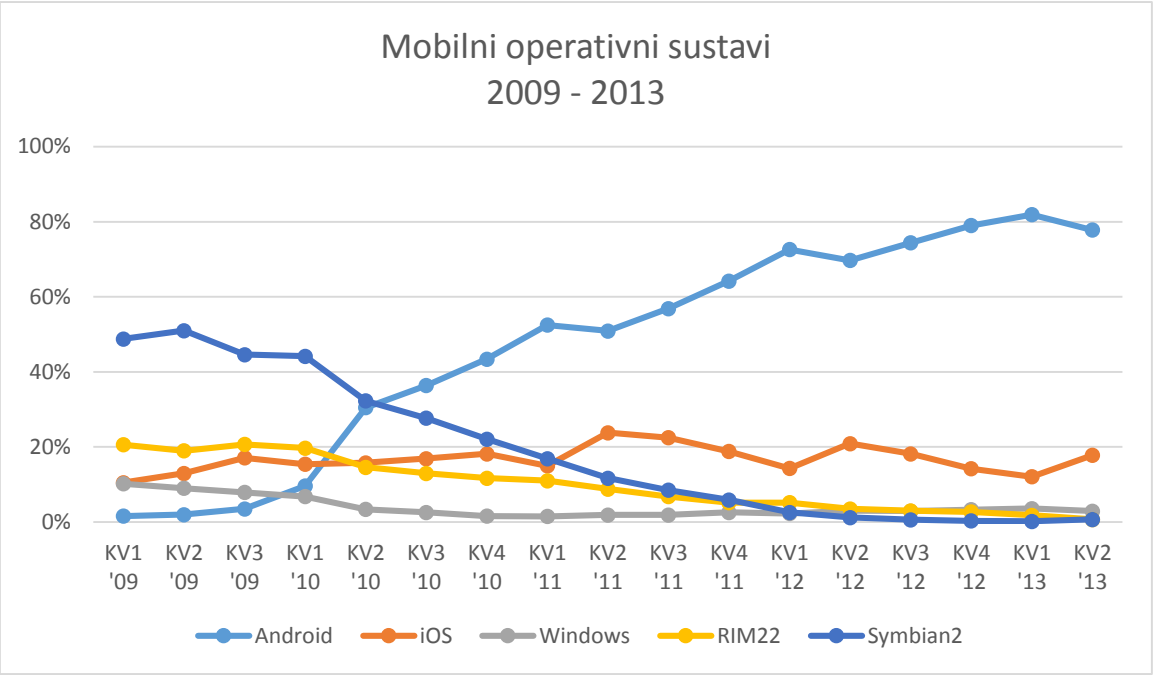
2. Razvoj softvera za Android

2.1. Android platforma

Android je slobodan operativni sustav otvorenog koda, kojeg razvijaju Google i Open Handset Alliance specifično za mobilne uređaje i koji se od objavljivanja 2008. proširio kao ni jedan drugi. Osim već postojećih milijardu uređaja[3] sa Androidom se svakodnevno aktivira još 1.5 milijuna novih uređaja[4]. Cilj održavanja otvorene platforme je zajedničko prilagođavanje i poboljšavanje sustav od strane različitih razvojnih programera umjesto samo jedne centralizirane kompanije. Linux jezgra operativnom sustavu daje dovoljnu fleksibilnost za prijenos na razne hardver platforme.

[3] Prema izjavi Sundar Pichai (dopredsjednik Google-a), Android je 1. rujna 2013. prešao milijardu kedinstvenih aktivacija: http://www.businessinsider.com/google-hit-1-billion-android-activations-2013-9

[4] Broj dnevnih aktivacija prema najnovijim podacima (travanj 2013.) statista.com: http://www.statista.com/statistics/278305/daily-activations-of-android-devices/

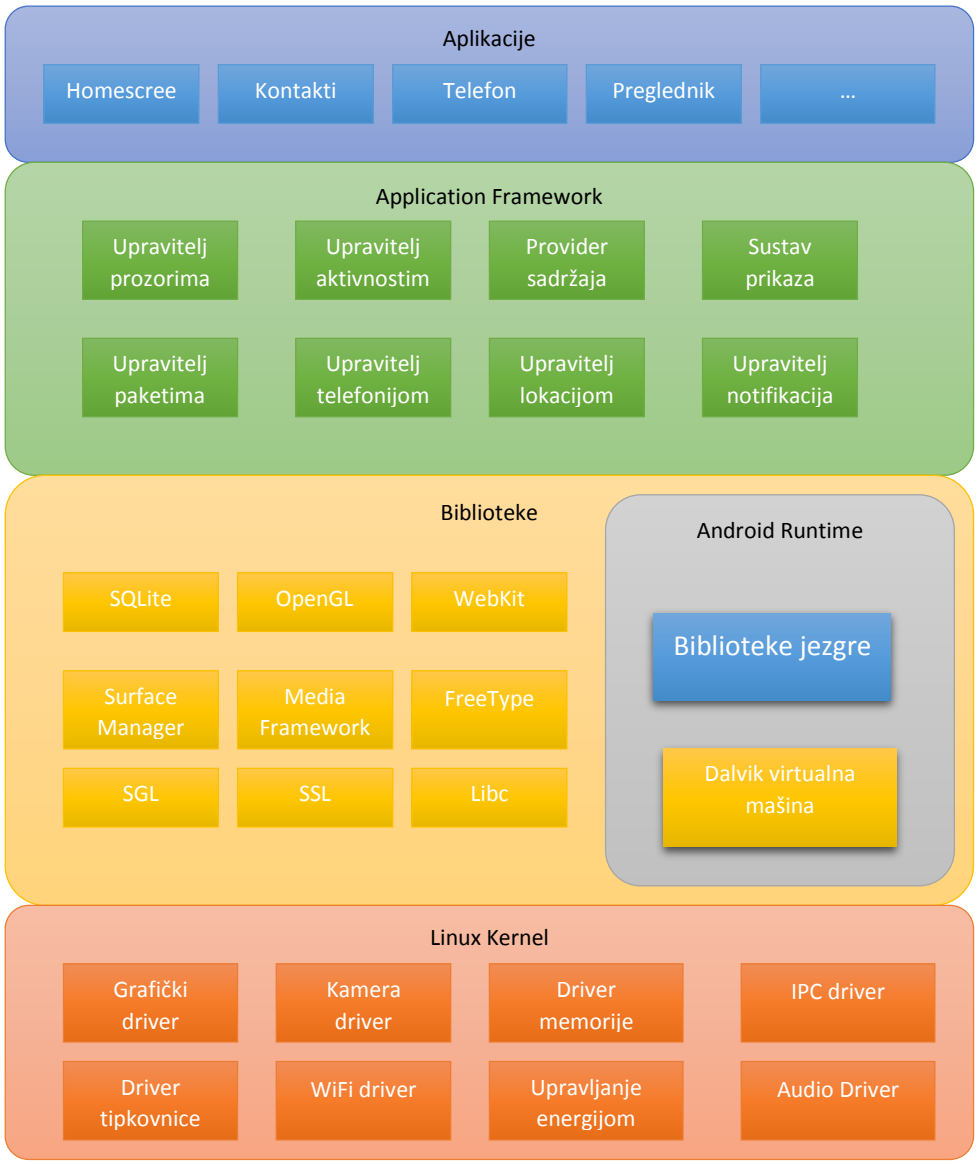


Slika 1 - Graf tržišne zastupljenosti mobilnih operativnih sustava[5]

2.2. Arhitektura

Sistemska arhitektura Android operativnog sustava podijeljena je na četiri sloja. Bazu Android arhitekture čini Linux jezgra 2.6, koji čini sloj između hardvera i ostatka arhitekture, te sadrži između ostalog upravljačke programe za hardver, upravljanje procesima i memorijom, te sigurnosno upravljanje. Android također uključuje niz biblioteka za nekoliko osnovnih komponenti, koje su radi boljih performansa napisane u C i C++. One omogućavaju veliki broj funkcija poput reprodukcije medija , 3D biblioteka, te standardna C biblioteka.

[5] Tržište mobilnih uređaja prema najnovijim podacima (travanj 2013.) statista.com: http://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/



Slika 2 – Arhitektura Android platforme

Svaka Android aplikacija pokreće se kao zaseban proces u vlastitoj Dalvik virtualnoj mašini. Ovo rješenje olakšava izradu aplikacija, pogotovo povećava sigurnost. Dalvik virtualna mašina razvijena je specifično za efikasno izvršavanje nekoliko instanci virtualnih mašina paralelno. Virtualna mašina izvršava datoteke u dex-formatu, koje su optimizirane za minimalnu potrošnju memorije. Takve datoteke se dobivaju konvertiranjem gotovog koda kompajliranog u Javi dx-alatima. S obzirom da se Dalvik bazira na registrima umjesto na stogu, virtualnoj mašini je potrebno manje međukoraka za izvršavanje byte-koda na procesoru.

3. Razvojna okolina

3.1. Eclipse

Većina Android aplikacija napisano je u Javi. Eclipse je široko prihvaćeni razvojno alat, te nudi proširenja koja omogućavaju razvoj za Android, zbog čega je u ovom radu odabran Eclipse 4.2 zajedno sa Android Development Tool-om (ADT) kao razvojnu okolinu. Ova kombinacija omogućuje razvoj unutar Windows (8.1) ili OS X okruženja, te testiranje na virtualnim mašinama koje simuliraju razne Android uređaje.

3.2. Software Development Kit

Osim ADT-a potreban je i Android Software Development Kit (SDK) koji sadrži veliki broj razvojnih alata. Među ostalim SDK dolazi sa programom za analizu grešaka, raznim bibliotekama, emulatorom mobilnih uređaja, dokumentacijom i korisnim primjerima za osnovne aktivnosti.

3.3. Native Delvelopment Kit

Native Development Kit sadrži paletu razvojnih alata koja programeru daje mogućnost od programa koje su napisani u C++ napraviti build u obliku Android projekta te ih prevesti u izvršni kod. Glavni razlog korištenja C++ koda je ubrzanje. Ne može se reći da nativni kod uvijek implicira ubrzanje programa s obzirom da «loše» napisani dijelovi C++ koda u usporedbi sa «dobro» napisanim dijelovima Java koda mogu dati slabije rezultate. Prednosti C++ jezika (u pogledu na performanse i brzinu) možemo iskoristiti, kada su u pitanju radnje koje zahtijevaju prvenstveno procesor i memoriju. Iz prethodnog iskustva vidljivo je da ovakav pristup daje vidno bolje rezultate kod npr. obrade slike, točnije kod prepoznavanja teksta. Nativne biblioteke se uobičajeno nalaze u

odgovarajućoj putanji biblioteke gdje ih sustav zahtjeva. Ta putanja se smije samo čitati, te zbog toga nije korisna za našu biblioteku. NDK pomaže tako da tu biblioteku sprema u Application Package (APK) datoteku. U trenutku instalacije aplikacije na nekom uređaju sustav stvara «libs» mapu kojoj aplikacija ima pristup, dok je ostalima pristup zabranjen.

4. Osnovni pojmovi kartografije

kako bi terminologija bila jednoznačna, potrebno je u okviru rada definirati neke pojmove geografije i geoinformatike. Najvažniji pojam vjerojatno jesu geografske koordinate, koje služe definiranju pozicije na zemljinoj površini na jednostavan i jednoznačan način. Koordinate se u tom slučaju sastoje od dvaju brojeva, geografske širine i geografske dužine. Navodi o geografskoj dužini i širini biti će navedeni u mjernoj jedinici stupnjeva, pri čemu se geografska širina rasprostire između polova, a nula stupnjeva odgovara ekvatoru, dok svaki od polova odgovara 90 stupnjeva širina. Bitno svojstvo stupnjeva širine je konstantna udaljenost. Dva mjesta identične geografske dužine koja se u širini razlikuju za jedan stupanj udaljena su otprilike 111 kilometara. Za razliku od stupnjeva širine stupnjevi dužine nisu ekvivalentne distance. Nula stupnjeva geografske dužine odgovaraju kružnici koja prolazi londonski dio grada Greenwich. Kut između točke promatranja prema opisujućoj geografskoj dužini se opisuje prema zapadu ili istoku, te pri tome onda iznosi najviše 180 stupnjeva.

Koordinate se mogu navoditi općenito na dva načina. Jedan način je podjela stupnjeva na precizniju jedinice: minute i sekunde. Pri tome se područje između stupnja dužine ili širine jednoliko dijeli na 60 manjih dijelova, minute, a zatim njih na opet 60 dijelova. Tako bi se Geo koordinate za Rijeku mogli prikazati kao 45° 19′ 37″ S 14° 26′ 32″ I, a Buenos Aires kao 34°35′ 59″ S 58° 22′ 55″ Z. Alternativno se broj stupnjeva može izraziti u obliku realnog broja. Ovaj način zapisivanja je proširen u zrakoplovstvu, ali je i prijenos u informatiku logičan prvenstveno zato što oblik u stupnjevima, minutama i sekundama predstavlja problem kod zapisivanja u memoriju, pa se preferira decimalni zapis. Koordinate južno od ekvatora i zapadno od nultog meridijana se zapisuju kao negativni brojevi. U tom slučaju koordinate Rijeke zapisale bi se kao 45.32694, 14.44222, a one Buenos Aires kao -34.599722, -58.381944.

Pri određivanju pozicije bitno je uzeti u obzir da Zemlja nije savršena kugla. Zbog toga je za kartografiju bitno koristiti model zemlje koji realnom obliku zemljine kugle dolazi čim bliže. Kako bi se taj oblik čim bolje opisao koristi se uglavnom geoid. To je oblik sa jednolikom silom gravitacije na cijeloj površini. Karte svijeta imaju oblik elipsoida, no taj oblik se godinama mijenja i sve vise

precizira, jer napredak znanosti omogućuje sve preciznija mjerenja. Postoji veliki broj takvih elipsoida, a najznačajniji u Europi je vjerojatno Bessel ellipsoid[6].

Još jedan važan pojam unutar kartografije je georeferenciranje. Ono opisuje proces određivanja geografskih koordinata za promatranu točku. Često se koristi i engleski pojam *geocoding.* Postoji nekoliko načina referenciranja. Može se referencirati putem adresa, tako da danim podacima dodamo adresu, pri čemu adresa ne mora nužno biti poštanska adresa, može se dodati na primjer i izlazna točka autoceste unutar odgovarajućeg konteksta, koja onda određuje preciznu geo koordinatu u koliko postoji baza podataka koja sadrži takve podatke. Drugi način referenciranja bilo bi direktno dodavanja koordinata nekom skupu podataka. Taj način geocodinga se naziva *geotagging* i uobičajeno je kod slika ili članaka na internetu. Postoji i obrnuti pristup koji se na engleskom naziva *reverse geocoding* pri čemu programu dajemo koordinate, koji vraća sve reference koje su označene na toj lokaciji. U ovom radu koristiti će se većinom metoda geotagging, što u širem pogledu spada u područje augmented reality. „Augmented reality je direktni ili indirektni pogled na fizičko, realno okruženje čiji elementi su augmentirani (dodani) od strane računalno generiranog senzornog unosa kao zvuk, video , grafike ili GPS podaci.“ [Ref 1]

[6] Bessel ellipsoid je referencijalni ellipsoid kojeg je razvio Friedrich Wilhelm Bessel 1841. na temelju tadašnjih podataka o Europi, Rusiji i Indiji i koji se do dan danas koristi u prerađenoj verziji.

5. Vrste kartografskih sustava

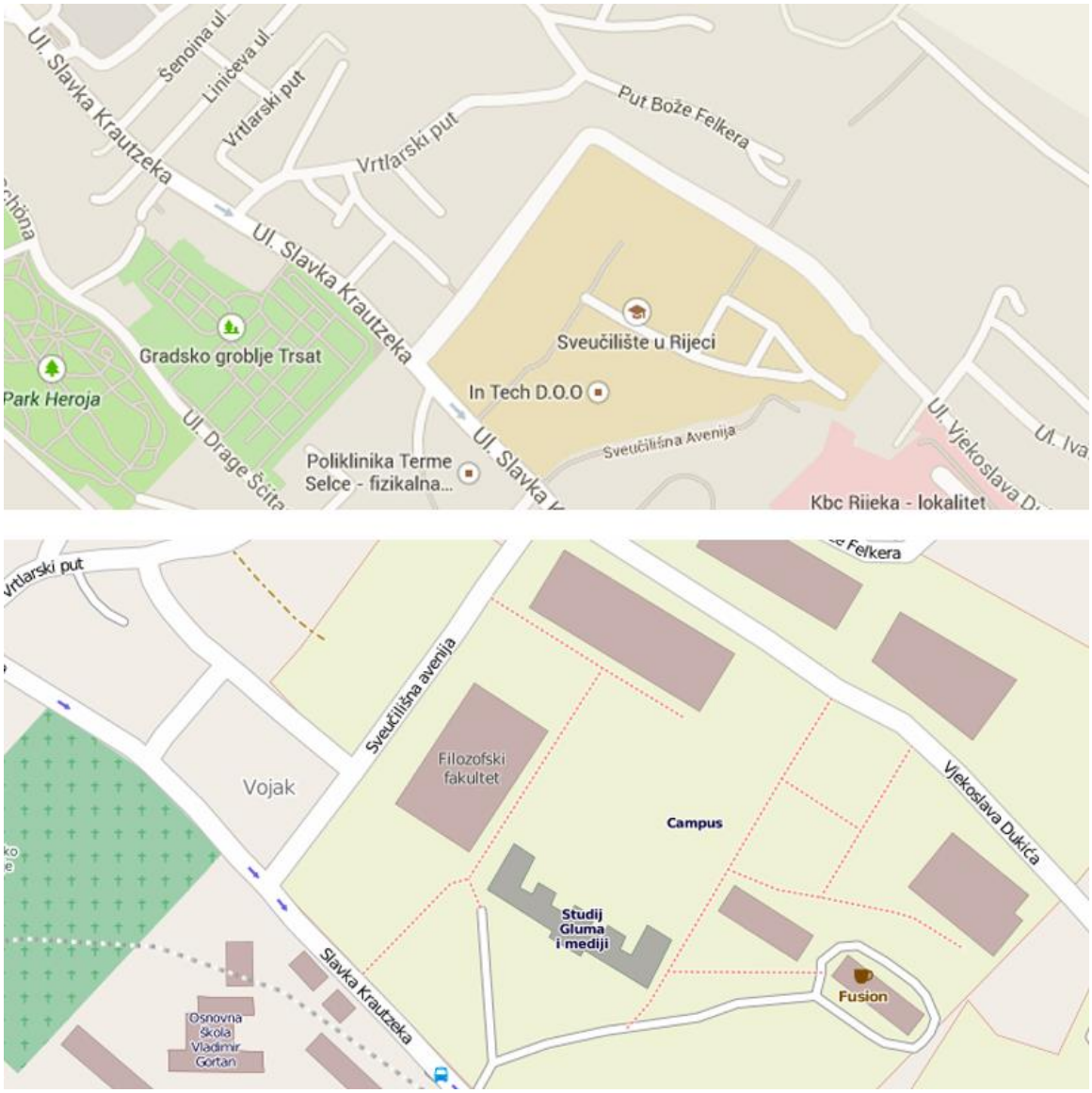
5.1. OpenStreetMap

OpenStreetMap je projekt sa ciljem stvaranja jedinstvene slobodne karte svijeta. Projekt je slobodan, otvoren i dobrovoljan i nastoji izgraditi što veću bazu podataka geografskih informacija. Podatke skupljaju dobrovoljci i objavljuju ih pod creative commons licencom, što korisniku omogućuje besplatno korištenja i ažuriranje materijala, čak uz vlastite izmjene istog.

OpenStreetMap sadržaji pohranjeni su u PostgreSQL bazi podataka, a za pristup njima dostupan nam je vlastiti API projekta, koji se trenutno nalazi u verziji 0.6. Pored toga OpenStreetMap nudi dva gotova rendera, koji nas opskrbljuju kartama. Mapnik je render koji daje karte u pixel formatu dok Osmarendere daje SVG format.

Osnovna struktura podataka u bazi podataka zabilježena je u obliku čvorova. Čvorovi opisuju otvorene i zatvorene puteve koji se vrjednuju kao elementi, te se dodatno mogu dovoditi u relaciju. Osim toga postoji mogućnost zapisivanja primjene tih čvorova unutar oznaka. Broj i vrsta takvih tagova u teoriji nije ograničen, iako u okviru projekta postoje pravila po kojima bi se elementi trebala kategorizirati. Sve informacije o kategorijama korisnik API-ja dobiva u odgovarajućoj dokumentaciji.

OpenStreetMap baza podataka sadrži ogroman broj informacija. Ulice se navode sa oznakom kategorije. Također sadrži biciklističke i pješačke staze, što ovaj projekt razlikuje od Google Mapsa, koji sadrži velikom dijelom samo puteve dostupne automobilima.



Slika 3 - Usporedba Google Maps (gore) i OpenStreetMaps (dolje) karata - Riječki kampus

Kao što je vidljivo na slici 3 razlika između ova dva izvora podataka može na lokalnoj razini biti dosta velika. Također vidimo da OpenStreetMaps ne skuplja samo podatke o ulicama nego i zgradama i bližoj okolini. Količina podataka naravno varira s obzirom na širinu sadržaja dostupnih na promatranoj točki.

Prikazani primjer ne treba dati dojam da su podaci od strane OpenStreetMaps kompletno na istom nivou ili bolji od konkurentnih izvora. S obzirom da su podaci skupljeni sa strane dobrovoljaca, kvaliteta i opširnost može biti različita od sredine do sredine, te je uglavnom ovisna o lokalnoj zajednici koja se bavi skupljanjem podataka.

Podaci skupljeni za OpenStreetMap projekt predstavljat će osnovu za aplikacije izrađene u okviru ovog rada, te licenca pod kojim su objavljeni isti daju korisniku više slobode u korištenju i omogućavaju offline spremanje karata što sa Google-ovim podacima nije moguće, a time predstavlja glavnu prednost naprema komercijalnom proizvodu Google Maps.

5.2. Cloudmade

Osnivač OpenStreetMaps-a Steve Coast zajedno sa Nick Blackom 2007. osniva kompaniju Cloudmade. Na temelju OpenStreetMap podataka firma nudi veliki broj sučelja, koji bi trebali, po vlastitoj izjavi, olakšati rukovanje geografskim podacima.

Paleta ponuđenih proizvoda je široka, te između ostalog sadrži Software Development Kit za iOS, niz alata za izradu navigacijskih aplikacija i vlastitu trgovinu za distribuciju licenci usluga sa geografskim podacima.

Pored iOS-a podržane su i druge mobilne platforme. Ponuđeni su razvojni alati za sve proširene platforme ka što su Java Micro Edition, Symbian i Android. Ovi razvojni alati nisu ponuđeni direktno od strane Cloudmade nego od trećih proizvođača.

Kako bi koristio Cloudmade, programer se može bez naknade registrirati. Nakon registracije dobiva jedinstveni ključ za API sa kojima autorizira upite na bazu podataka iz aplikacija. Ovisno o aplikacijama i vrsti pristupa programer može generirati više ključeva, a pristup sadržajima podliježe isto kao i kod OpenStreetMapsa također *creative commons* licenci. Ovo programeru omogućava slobodno širenje podataka ovisno o potrebama pod uvjetom da navodi izvornog autora i objavi aplikaciju pod odgovarajućom licencom.

Nadalje Cloudmade nudi API kojim možemo direktno putem http protokola pristupiti određenim podacima. U to spada i mogućnost skidanja dijelova karte svijeta u obliku bitmape. Upit pri tome ima uvijek slijedeći oblik:

```
http://a.tile.cloudmade.com/KLJUC/STIL/VELICINA/C/A/B.png
```

a.tile.cloudmade.com je server, pri čemu se prvo slovo može mijenjati u koliko su nam potrebni simultani upiti. Ovo omogućava nekoliko konekcija u isto vrijeme. Nakon toga slijedi API ključ, kojeg smo dobili pri registraciji. Zatim slijedi vrijednost 1 ili 2 što označava optički izgled karte koju povlačimo dok za veličinu možemo definirati 64 ili 256 piksela (ista vrijednost za širinu i za visinu). Zadnje tri varijable označavaju okvir unutar kojeg povlačimo podatke i definiran je na isti način kao i kod OpenStreetMapsa.

Cloudmade također omogućava skidanje karata u vektorskom formatu. Treba uzeti u obzir da je datoteka koja rezultira ovakvim upitom u svg formatu, koji se nativno ne može prikazati unutar Androida, pa samo po sebi ne predstavlja alternativu navigacijskom softveru.

Još jedna od ključnih funkcija http-API-ja je mogućnost zahtjeva koji računa rute. Takav upit bio bi u sljedećem obliku:

```
http://routes.cloudmade.com/KLJUC/api/0.3/POCETAK,[PRIJELAZ...],KRAJ/TIP.
FORMAT
```

Pomoću POCETAK, PRIJELAZ i KRAJ definiraju se respektivno početna točka, međutočke i odredište rute (uvijek u obliku STUPANJ ŠIRINE, STUPANJ DUŽINE). Definiranjem vrijednosti TYPE biramo prijevozno sredstvo pri čemu Cloudmade podržava pješačke biciklističke i automobilske rute. Rezultat je vraćen u JSON ili GPX format ovisno o tome što smo definirali kao FORMAT vrijednost. Pored međutočaka koje su opcionalne, mogu se još definirati jezik, mjerne jedinice udaljenosti i za JavaScript upite eventualna Callback funkcija kojoj se rezultat treba predati.

Cloudmade također daje mogućnost ranije navedenog geocodinga u obliku http zahtjeva. Za izvršavanje takvog zahtjeva potrebno je kao parametar poslati u obliku adrese (npr. “Ilica, Zagreb”) ili u obliku opisa (npr. “Zagrebačka katedrala”). Provjera u ovom slučaju izgleda kako slijedi:

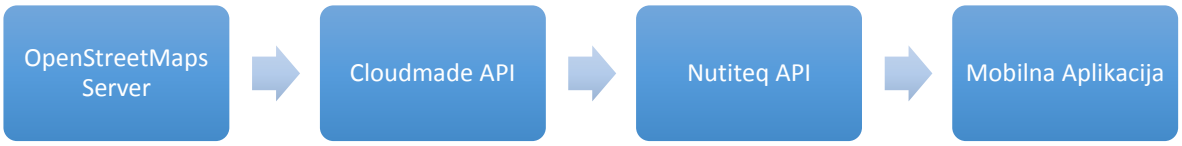
```
http://geocoding.cloudmade.com/KLJUC/geocoding/v2/find.FORMAT?query=UPIT&
PARAMETRI
```

Kao povratni format predviđeno je JSON ili HTML. Opis adrese spremamo u QUERY, pri čemu se nekoliko riječi spaja sa znakom zbrajanja. Kao PARAMETRI može se između ostalog navesti područje pretraživanja. Istim zahtjevom možemo iskoristiti jedan izuzetno koristan način pretraživanja definirajući varijablu object type: Takvim upitom možemo pronaći objekte određenog tipa u okolini definirane točke.

Predstavljeni upiti omogućili bi nam bez problema integraciju u Android aplikaciju, ali s obzirom da Nutiteq API nudi još čitav niz mogućnosti specifično za Android, koristiti ćemo njega za razvoj primjera unutar rada.

5.3. Nutiteq Mobile Map API

Estonska kompanija Nutiteq razvila je API koji koristi podatke Cloudmade-a. Tako imamo pristup klasama koji znatno olakšavaju učitavanje dijelova karata, te odmah donose odgovarajuće View objekte i time omogućavaju interakciju s kartom. Nadalje nudi interface za izvedbu izračuna rute koji se onda može grafički prikazati, kao i ugrađeni servis za geocoding upite.



Slika 4 - Put povratne informacije nakon zahtjeva

Za prikaz karata možemo koristiti pred definiranu funkcionalnost prikaza dodatnih informacija. Tako možemo dodati lokacije, rute i poligone. Na taj način možemo vizualno realizirati upute ili ucrtane turističke atrakcije.

Nutiteq distribuira svoj API pod GNU General Public License (GPL), ali i pod komercijalnim licencama. Ukoliko se programer odluči objaviti aplikaciju pod GPL licencom također dobiva GPL za Maps API i time ima pristup izvornom kodu.

Nutiteq API ne ograničava rad samo na Cloudmade, već je moguć rad sa drugim kartografski uslugama pomoću postojećih klasa čiji broj raste, ali sada već omogućavaju direktni rad sa Mapnik OpenStreetMap renderom.

Unutar okvira ovog rada koristit će se Nutiteq za crtanje ruta između dviju točaka koje korisnik bira odnosno dodirom odabire. Nututeq se većinom bazira na Cloudmade API-ju, ali je posebno zanimljiv zbog potpune kompatibilnosti sa Mapsforgom na kojem se neke klase za rendering lokalno spremljenih klasa baziraju.

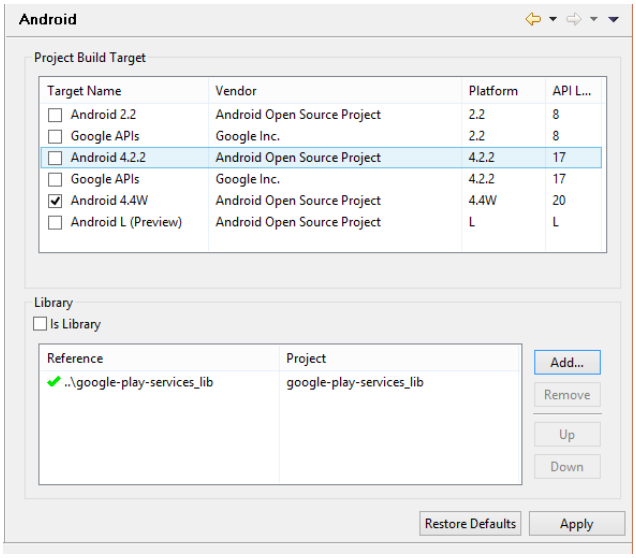
6. Primjena kartografskih sustava

6.1. Google Maps API v2

Sljedeće poglavlje služi kao uvod u primjenu kartografskih sustava unutar android aplikacija. Primjer aplikacije biti će izrađen pomoću Google Maps API v2,s obzirom da je upravo Google postavio tržišni standard, te je od svih dostupnih okruženja najstabilniji i nudi sve osnovne funkcionalnosti. Također će se ukazati i na ograničenja ovog okruženja.

6.1.1. Postavljanje razvojne okoline

Kako sve potrebne biblioteke dolaze kao dio Google Play sevisa, potrebno je prije kreiranja novog projekta u Eclipse dodati Android SDK i pomoću njega skinuti Google Play Services SDK. Kako bi aplikacija tijekom kompajliranja uključila navedene biblioteke, potrebno je u build target dodati play services (prikazano na slici X). Također moramo navesti Play biblioteke u Android manifestu datoteke.



Slika 5 – Dodavanje Google Play biblioteke u build path

Za pristup Google Maps serverima potrebno je kreirati jednoznačan ključ koji nam omogućava pristup neograničenom broju korisnika, ali koji svaki zahtjev jednoznačno identificira. Ključ kreiramo

na Google API konzoli, gdje registriramo aplikaciju i njihove certifikate, te zahtijevamo jedan ili više ključeva i uključimo ga u Android manifest (Slika X).

```
<meta-data
                android:name="com.google.android.maps.v2.API_KEY"
                android:value="AIzaSyDq9BahdzL-9EQPVBbu08JTSYJWRgG7PrE"/>
```

Isječak izvornog koda 1

Potrebno je dodati dozvole za pristup internetu, provjeru mreže, lociranju pomuću WiFi i GSM, lociranje pomoću GPS-a i za zapisivanje u vanjsku memoriju.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Isječak izvornog koda 2

Za rendering Google Maps koristi OpenGL ES version 2, pa ga treba obavezno uključiti u manifest svakog projekta.

```
<uses-feature
                 android:glEsVersion="0x00020000"
                 android:required="true"/>
```

Isječak izvornog koda 3

Nakon što su sve potrebne dozvole dodane u manifest, potrebno je dodati Map View u main.xml datoteku. Unutar te datoteke definira se frontend aplikacije, odnosno svi elementi korisničke interakcije poput gumba, prozora unutar kojeg je karta vidljiva i slično. Za ovaj primjer ćemo dodatno uključenu kartu centrirati varijablama map:cameraTargetLat i map:cameraTargetLng (definira početnu točku perspektive odnosno „kamere“). Primjera radi, karta će biti centrirana u Rijeci.

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
          xmlns:map="http://schemas.android.com/apk/res-auto"
          android:id="@+id/map"
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:name="com.google.android.gms.maps.SupportMapFragment"
          map:mapType="normal"
          map:cameraTargetLat="45.326418"
          map:cameraTargetLng="14.447845"/>
```

Isječak izvornog koda 4

6.1.2. Main Activity

Unutar MainActivity.java napravljena je javna klasa MainActviy koja nasljeđuje Activity klasu, odnosno u slučaju rada sa Maps API, Google preporučuje naslijediti FragmentActivity.

```
public class MainActivity extends FragmentActivity {
   private GoogleMap mMap;
   private final LatLng LOCATION_KAMPUS = new LatLng(45.327783, 14.466744);
   private final LatLng LOCATION_CENTAR = new LatLng(45.326418, 14.447845);
}
```

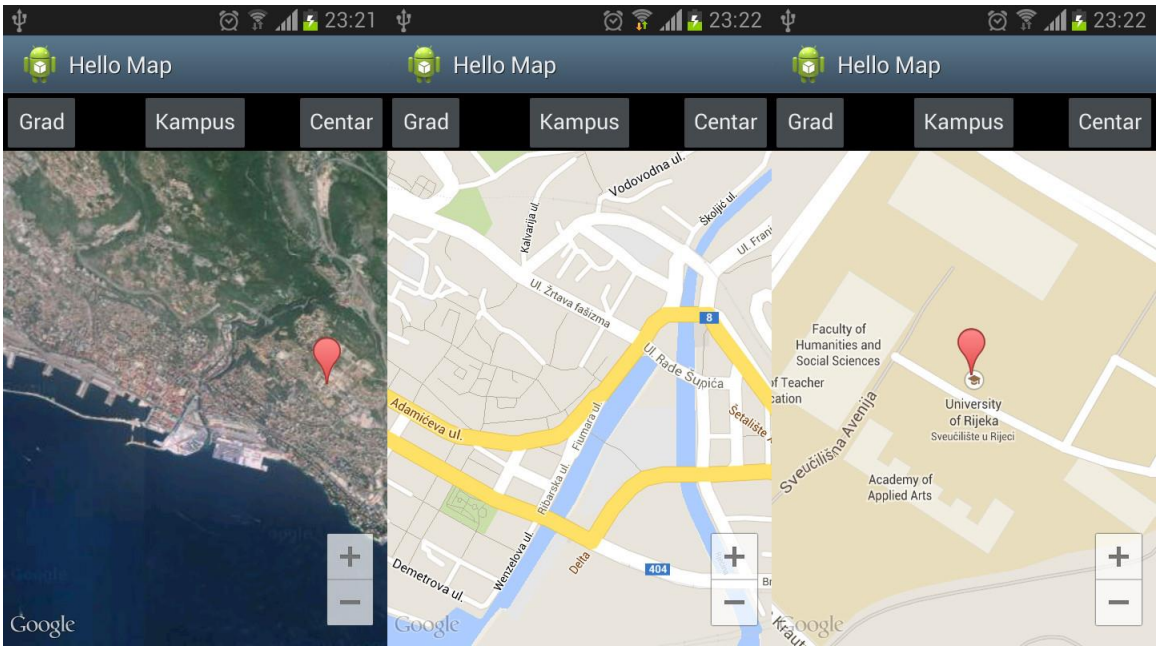
Isječak izvornog koda 5

Dvije konstante definirane u prethodnom dijelu koda potrebne su za aplikaciju koja služi kao primjer za ovo poglavlje. Aplikacija se sastoji od četiri pregleda. Početni ekran aplikacije centrira kartu na Rijeku, radi čega je bilo potrebno definirati geografske koordinate centra Rijeke. Na vrhu aplikacije nalazi se tri gumba koja mijenjaju pogled karte na Centar Rijeke, kampus sveučilišta i prikaz cijelog grada.

Gumbe je potrebno definirati u ranije spomenutom main.xml layout-u i dodati im onClick funkciju, koju možemo pozvati iz glavne aktivnosti.

```
public void onClick_Grad(View v) {

	mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
	CameraUpdate update = CameraUpdateFactory.newLatLngZoom
	(LOCATION_CENTAR, 13);
	mMap.animateCamera(update);
                                    }
public void onClick_Kampus(View v) {

	mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
	CameraUpdate update = CameraUpdateFactory.newLatLngZoom
	(LOCATION_KAMPUS, 18);
	mMap.animateCamera(update);

                                    }
public void onClick_Centar(View v) {

	mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
	CameraUpdate update = CameraUpdateFactory.newLatLngZoom
	(LOCATION_CENTAR, 16);
	mMap.animateCamera(update);

                                    }
```

Isječak izvornog koda 6



Slika 6 – Slike ekrana u 3 moguća stanja aplikacije

6.2. Mapsforge

Mapsforge je biblioteka za rendering vektorskih mapa, koje zauzimaju znatno manje prostora od rasterskih mapa (npr. MBTiles format). Kvaliteta renderinga i brzina je nešto manja kada se renderira na uređaju nego kada se ti procesi obavljaju na serverskoj strani, kao što je to praksa kod Google Mapsa.

Najveća prednost kod takvog pristupa je da se mogu lokalno spremiti karte te nakon prvog skidanja više nije potrebna podatkovna veza. Koliko god je u današnje vrijeme uobičajeno imati podatkovni plan uz pametni telefon, ovakve aplikacije bile bi korisne primjerice turistima koji se nalaze u roamingu i nemaju stalno pristup internetu.

6.2.1. Lokalna karta

Prvi zadatak je, znači, generirati kartu u ispravnom formatu za iščitavanje, u ovom slučaju .map format. Mapsforge biblioteka sadrži dodatak mapsforge-writer za Osmosis[7]. Ovaj dodatak podržava razne izvore, ali za ovaj primjer najjednostavnije je direktno sa web stranice sa servisa Cloudmade eksportirati kartu u osm formatu, te sa web stranice Openstreetmap.org preuzeti koordinate rubnih točaka karte. Selektirana je karta Rijeke, i konvertirana sa sljedećom naredbom, pri čemu varijabla *bbox* označava rubne točke karte

```
osmosis --read-xml file=c:\croatia-latest.osm --mapfile-writer
file=c:\rijeka.map bbox=14.2548,45.2763,14.6036,45.4079
```

[7] **Osmosis** je Java aplikacija za naredbeni redak koja služi za obradu OSM podataka. Alat se sastoji od raznih sučelja koja zajedno izvršavaju veće zadatke, primjerice čitanje i zapisivanje iz baza podataka ili drugih podatkovnih izvora.

Dobivenu kartu potrebno je dodati u *assets* mapu unutar strukture projekta, te će se kasnije definirati unutar glavne aktivnosti lokacija datoteke.

6.2.2. Glavna aktivnost

Za početak slijedi ključna inicijalizacija aplikacije, koja omogućuje pozive prema biblioteci i prikuplja podatke o uređaju poput rezolucije ekrana, koje su potrebne za što bolji rad renderinga.

```
AndroidGraphicFactory.createInstance(this.getApplication());
```

Isječak izvornog koda 7

Layout datoteka u xml formatu izgleda skoro identično kao layout Google Maps aplikacije. Isto kao kod prvog primjera potrebno je rezervirati prostor za MapView prozor. Poslije toga može se kreirati prva i glavna aktivnost, koja počinje sa mapview inicijalizacijom i dodatnom definicijom početne pozicije perspektive, razine zoom-a i sl.

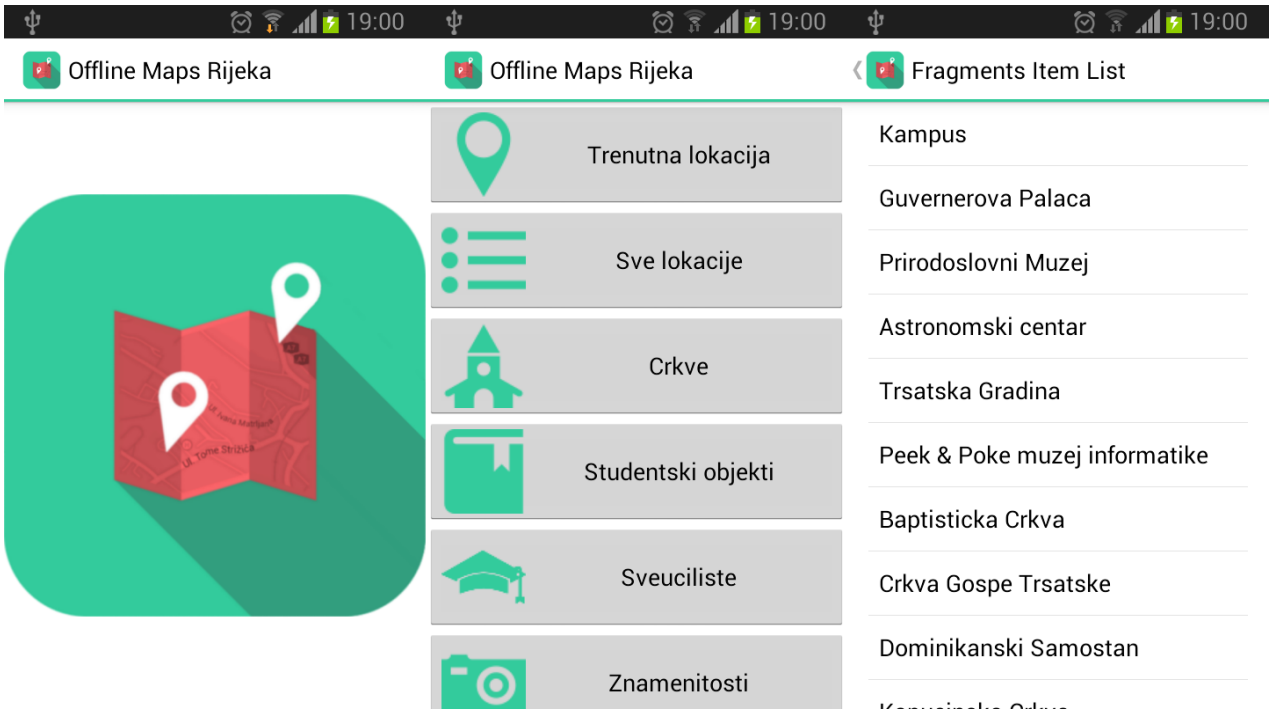
```
this.mapView = new MapView(this);
setContentView(this.mapView);
this.mapView.setClickable(true);
this.mapView.getMapScaleBar().setVisible(true);
this.mapView.setBuiltInZoomControls(true);
this.mapView.getMapZoomControls().setZoomLevelMin((byte) 10);
this.mapView.getMapZoomControls().setZoomLevelMax((byte) 20);
```

Isječak izvornog koda 8

6.2.3. Znamenitosti

Aplikacija koja se izrađuje kao primjer sastojati će se od nekoliko različitih pregleda. Nakon „splash screen-a“ koji prokazuje logotip aplikacije dok se ona ne učita, otvara se izbornik unutar kojeg se mogu pokrenuti glavne funkcije aplikacije. Prvi gumb („Moja lokacija“) otvara kartu centriranu na

trenutnoj lokaciji (dohvaćene pomoću GPS senzora), zatim kategorije znamenitosti ili točaka interesa („Crkve“, „Studentski objekti“, „Sveučilište“ i „Znamenitosti“) koje otvaraju pregled karte centrirano na centru grada s markerima na odgovarajućim lokacijama. Zadnja opcija u izborniku se zove „Sve lokacije“ i otvara listView popis svih lokacija iz svih kategorija i koje se opet dodirom mogu otvoriti, odnosno prikazati na karti. Svaka od kategorija ima zasebnu klasu sa odgovarajućim konstruktorom niza unutar kojeg će se spremati podaci o točkama. Kao primjer prikazan je izvorni kod klase „Sveuciliste“ koji prikazuje definiciju klase, dodavanje instanci (u ovom slučaju objekti Sveučilišta u Rijeci) i poziv na pregled karte.



Slika 7 – Slika ekrana aplikacije. „Splash screen“ (lijevo), Izbornik (sredina) i „Sve lokacije“ (desno)

```
public class Sveuciliste {
    /**
     * SveucilisteItem klasa sadrzi informacije, poput lokacije, opisa, id
itd.
     */
    public static class SveucilisteItem {
        public final String content;
        public final String id;
        public final LatLong location;
        public final String text;

        public SveucilisteItem(String id, String content, LatLong location,
                               String text) {
            this.id = id;
            this.content = content;
            this.location = location;
            this.text = text;}
    }
    /**
     * Kreiranje nove liste lokacija
     */
    public static final List<SveucilisteItem> SVEUCILISTE = new Ar-
rayList<SveucilisteItem>();

    static {
        addItem(new SveucilisteItem("1", "Akademija primijenjenih
umjetnosti", new LatLong(45.327105, 14.466282), "Akademija primijenjenih
umjetnosti"));
        addItem(new SveucilisteItem("2", "Ekonomski fakultet", new Lat-
Long(45.331617, 14.434742), "Ekonomski fakultet"));
        addItem(new SveucilisteItem(
                "3",
                "Fakultet za menadzment",
                new LatLong(45.303260, 14.282124),
                "Fakultet za menadzment"));
        addItem(new SveucilisteItem("4", "Filozofski fakultet", new Lat-
Long(45.327955, 14.465712), "Filozofski fakultet"));
        addItem(new SveucilisteItem("5", "Gradevinski fakultet", new Lat-
Long(45.328954, 14.467807), "Gradevinski fakultet"));
        addItem(new SveucilisteItem("6", "Medicinski fakultet", new Lat-
Long(45.335873, 14.429459), "Medicinski fakultet"));
        addItem(new SveucilisteItem("7", "Odjel za Informatiku", new Lat-
Long(45.328657, 14.466715), "Odjel za Informatiku"));
        addItem(new SveucilisteItem("8", "Pomorski fakultet", new Lat-
Long(45.330504, 14.436122), "Pomorski fakultet"));
        addItem(new SveucilisteItem("9", "Pravni fakultet", new Lat-
Long(45.344051, 14.416549), "Pravni fakultet"));
    }
    private static void addItem(SveucilisteItem item) {
        SVEUCILISTE.add(item);
        SVEUCILISTE_MAP.put(item.id, item);
    }
}
```

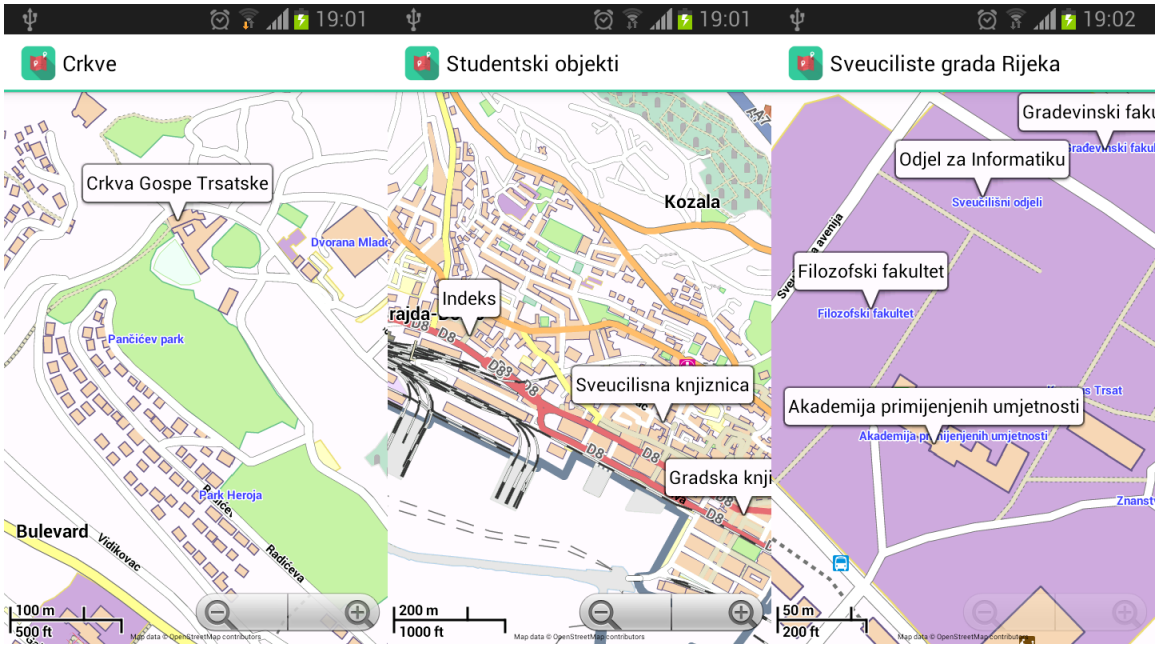
Isječak izvornog koda 9

6.2.4. Markeri

Sve što treba napraviti nakon što su lokacije znamenitosti definirane, je u ranije kreiranoj glavnoj aktivnosti dodati funkciju koja prikazuje markere za sve definirane točke, ali kartu uvijek centrira na odabranu točku. U tu svrhu definira se instanca tipa *bitmap* kako slijedi. Za ovakve slučajeve mapsforge nudi predefinirane klase Bubble i bubbleView koje funkcioniraju na sličan način kao Google Maps Utils. Pomoću ovih klasa moramo napraviti novi marker tipa bubble i dodijeliti mu karakteristike pozivom funkcije bubbleView.

```
private Bitmap bubble;

	@Override
	protected void createLayers() {
		super.createLayers();
		for (Sveuciliste.SveucilisteItem item : Sveuciliste.SVEUCILISTE)
{
			TextView bubbleView = new TextView(this);
			Utils.setBackground(
					bubbleView,
					getResources().getDrawable(

	R.drawable.balloon_overlay_unfocused));
			bubbleView.setGravity(Gravity.CENTER);
			bubbleView.setMaxEms(20);
			bubbleView.setTextSize(15);
			bubbleView.setTextColor(Color.BLACK);
			bubbleView.setText(item.text);
			bubble = Utils.viewToBitmap(this, bubbleView);
			bubble.incrementRefCount();
			this.layerManagers
				.get(0)
				.getLayers()
				.add(new Marker(item.location, bubble, 0, -bubble
				.getHeight() / 2));
		}
	}

	@Override
	protected void createMapViewPositions() {
		super.createMapViewPositions();
		this.mapViews.get(0).getModel().mapViewPosition
				.setCenter(Sveuciliste.SVEUCILISTE.get(1).location);
	}

	@Override
	protected void destroyLayers() {
		bubble.decrementRefCount();
	}

	@Override
	protected void onStart() {
		super.onStart();
		this.mapViewPositions.get(0).setCenter(
				Sveuciliste.SVEUCILISTE.get(1).location);
	}
```

Isječak izvornog koda 10



Slika 8 – Slike ekrana aplikacije centrirane na odabranim lokacijama / Markeri sa dodatnim informacijama o znamenitostima

Kada se izgradi aplikacija kako je opisano ranije, dobivena aplikacija prikazana je na slici 6. Ta aplikacija ima iste funkcionalnosti kao ranije kreirana Google Maps aplikacija, čak i interakcije s korisnikom je skoro potpuno ista. Pokreti s više prstiju (na ekranima na dodir), podudaraju se u obje okoline. Sadržaja karata se minimalno razlikuju, a mapsforge nudi mogućnost lokalnog renderinga i lokalnog spremanja karata što je velika prednost. Brzina renderinga ovisi o brzini uređaja, ali većina suvremenih uređaja ispunjava zahtjeve framework-a i obavlja sve radnje bez daljnjih problema.

Unutar okvira mapsforge okruženja također može izvršavati više aktivnost tipa mapView u isto vrijeme, tako da podijelimo ekran na dva dijela. Također možemo postaviti markere bilo koje vrste i crtati poligone na karti. Ono što mapsforge ne sadrži, jest biblioteka koja bi riješila problem navigacije ili računanja ruta. Da bi ostvarili osnovne funkcije navigacije potrebno je posegnuti za Nutiteq bibliotekama.

6.3. Nutiteq

6.3.1. Računanje ruta

Kako je lokalni rendering karata prilično zahtjevno po pitanju resursa, najlakši način realizirati računanje ruta unutar aplikacije je prikupiti početnu i završnu točku, proslijediti ih serveru i odgovor iscrtati na karti opet lokalno. MapQuestDirections Open routing API8 omogućava upravo ovaj zadatak. U tu svrhu se kreira klasa pod nazivom MapQuestDirections.

U prvoj instanci aplikacije korisnik vidi kartu i Map View koji je izrađen u prethodnom poglavlju. Kada korisnik prvi puta "dodirne" kartu, na tom mjestu se treba iscrtati zeleni marker koji označava početnu točku rute. Drugi dodir stvara crveni marker koji označava završnu točku rute. Nakon toga podaci se pomoću API zahtjeva pošalju. Odgovor na zahtjev je najkraći put od točke A do točke B (radi se o navigaciji za automobil, pješačke i biciklističke staze ne dolaze u obzir), koji se iscrtava na karti i na svakom skretanju stavlja strjelicu koja označava smjer skretanja. Simboli skretanja su u obliku url putanje, te nisu lokalno spremljeni.

```
// stvaranje markera pomocu R.drawable
        Bitmap olMarker = UnscaledBitmapLoader.decodeResource(getResources(),
                R.drawable.olmarker);
        StyleSet<MarkerStyle> startMarkerStyleSet = new
StyleSet<MarkerStyle>(

MarkerStyle.builder().setBitmap(olMarker).setColor(Color.GREEN)
                .setSize(MARKER_SIZE).build());
        startMarker = new Marker(new MapPos(0, 0), new DefaultLabel("Start"),
                startMarkerStyleSet, null);

        StyleSet<MarkerStyle> stopMarkerStyleSet = new StyleSet<MarkerStyle>(
                MarkerStyle.builder().setBitmap(olMarker).setColor(Color.RED)
                .setSize(MARKER_SIZE).build());
        stopMarker = new Marker(new MapPos(0, 0), new DefaultLabel("Stop"),
                stopMarkerStyleSet, null);
```

Isječak izvornog koda 11

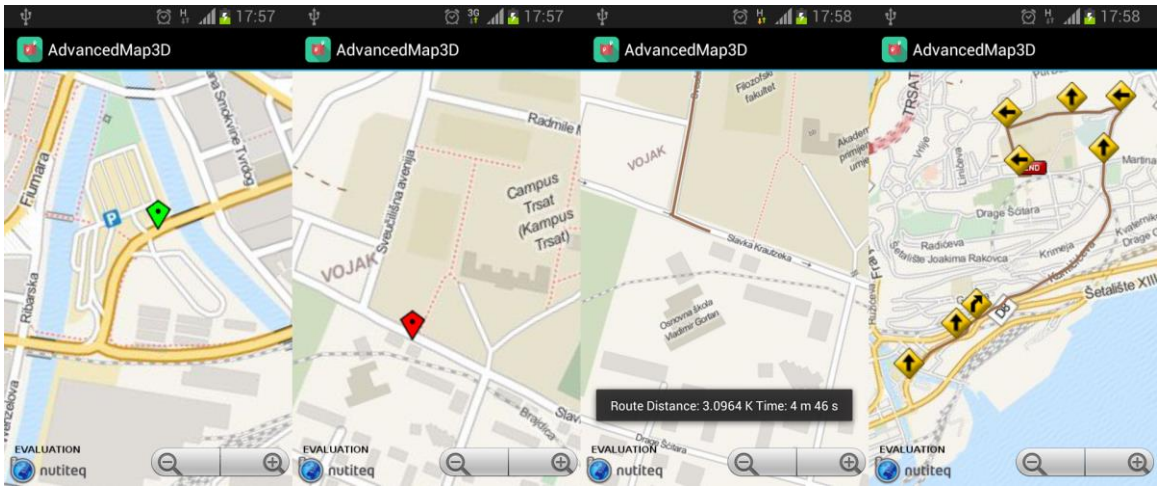
Zahtjev nije potrebno ponovno pisati, potrebno je jedino uključiti paket *com.nutiteq.services.routing.MapQuestDirections* koji sadrži odgovarajuću funkciju kojoj

8

proslijedimo: koordinate početne i završne točke, dodatne opcije rute (npr. „fastest“) i jedinstveni ključ za MapQuest API koji identificira projekt.

```
public void showRoute(final double fromLat, final double fromLon,
            final double toLat, final double toLon) {

        Log.debug("calculating path " + fromLat + "," + fromLon + " to "
                + toLat + "," + toLon);

        Projection proj = mapView.getLayers().getBaseLayer().getProjection();

        StyleSet<LineStyle> routeLineStyle = new
StyleSet<LineStyle>(LineStyle.builder().setWidth(0.05f).setColor(0xff9d7050).
build());
        Map<String, String> routeOptions = new HashMap<String,String>();
        routeOptions.put("unit", "K"); // K - km, M - miles
        routeOptions.put("routeType", "fastest");

        directionsService = new MapQuestDirections(this, new MapPos(fromLon,
fromLat), new MapPos(toLon, toLat), routeOptions, MAPQUEST_KEY, proj,
routeLineStyle);
        directionsService.route();
    }
```

Isječak izvornog koda 12



Slika 9 - Slike ekrana aplikacije / 4 moguća stanja aplikacije

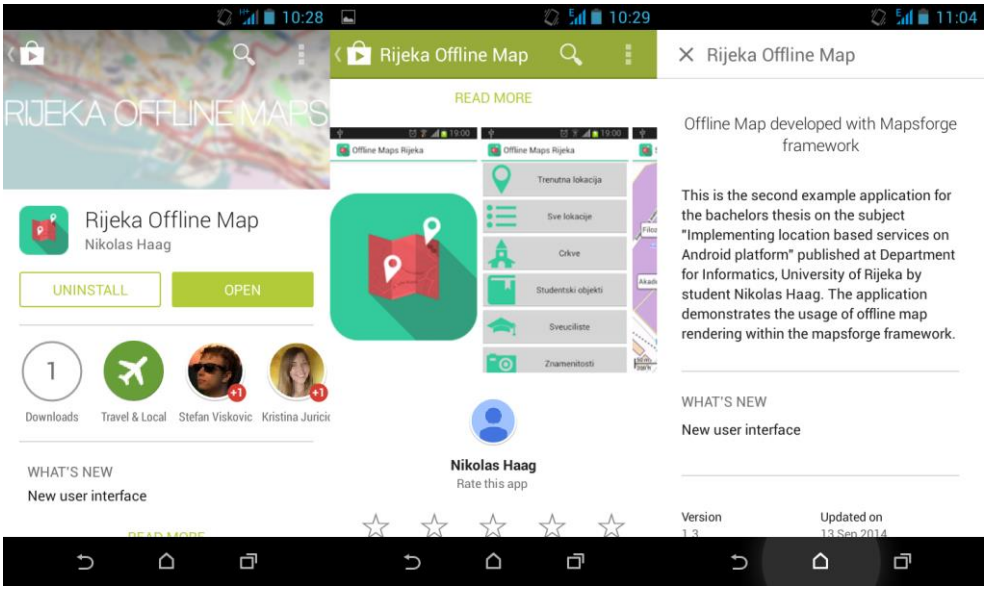
7. Objava Aplikacije

Registrirani android developeri imaju mogućnost objave svojih aplikacija u Google Play trgovini, nakon čega postaju dostupne na skoro svakom android uređaju, s obzirom da Play trgovina dolazi predinstalirano na većini uređaja. Prvi korak je registracija za Google Play objavljivački račun.

Registriracija se obavlja kroz Google Play Developer Console gdje se unose osnovni podaci - ime developer, email adresa i drugo. Zatim slijedi ugovor pod nazivom *Developer Distribution Agreement* koji se odnosi na regiju ili zemlju developera. Aplikacije koje se objavljuju na Trgovini Google Play moraju se pridržavati politike Developer pragrama[9] i američkog zakona o izvozu.

Potvrda o verifikaciji prijave dobiva se putem email adrese koja je prijavljena. Nakon registracije i dobivanja verifikacijskog emaila, moguća je prijava u Google Play Developer Console što je ujedno i početna stranica za objavljivanje aplikacija na trgovini. Nakon odabira aplikacije koja se zeli obaviti, u padajućem izborniku se mijenja status iz "Ready to Publish" u "Publish the App". Dodaje se opis aplikacije i barem po jednu sliku ekrana za mobilnu verziju i za tablete. Nakon što je aplikacija objavljena u trgovini, možemo objavljivati novije verzije programa, pri čemu treba pripaziti da u AndroidManifest.xml datoteci povećamo atribute versionCode i versionName. VersionCode predstavlja verziju aplikacije, dok se u versionName sprema oznaka verzije koja je vidljiva krajnjem korisniku.

[9] Prihvaćanjem Developer programa developeri se slažu da neće distribuirati aplikacije koje sadrže nasilje, širenje mržnje, igru na sreću i druge sadržaje. Detalji se mogu pročitati na sljedećem linku (zadnja provjera 15.9.2014.): https://play.google.com/about/developer-content-policy.html



Slika 10 – Prikaz objavljene aplikacije u Trgovini Play

8. Zaključak

Može se reći da je moguće riješiti postavljeni zadatak odnosno implementirati kartografske sustave u okvirima android aplikacija. Hardver suvremenih uređaja[10] ispunjava sve uvjete za izvršavanje svih funkcionalnosti navedenih primjera, uključujući zahtijevanje ruta od internetskih poslužitelja treće strane. Mogućnost zahtijevanja trenutne pozicije pomoću GPS modula dozvoljava rad sa podacima u realnom vremenu.

Funkcije suvremenih android uređaja, naročito ekrani na dodir, ali i sam operativni sustav omogućavaju jednostavno i intuitivno korištenje aplikacija na području kartografije. Zbog toga se na područjima kartografije, navigacije i *augmented reality* može u budućnosti očekivati sve veći broj implementacija. Kako se pametni telefoni sve više integriraju u svakodnevni život, potencijalnih primjena kartografskih sustava je bezbroj. Navedeni primjeri u radu trebaju biti tek uvod u obrađene biblioteke, kako bi se ukazalo na potencijal i mogućnosti u razvoju ove vrste mobilnih aplikacija.

Eclipse IDE i dane biblioteke omogućavaju suvereni razvoj aplikacija bez previše predznanja. Većina biblioteka je dobro dokumentirano i dozvoljava bezbolan uvod za programere koji se prvi put susreću sa implementacijom kartografskih sustava na android platformi. Jedina točka kritike za razvojni alata je emulator android uređaja koji koristi previše resursa, te je testiranje aplikacija dosta sporo u odnosu na prave uređaje. Također simulacija senzora u trenutku pisanja ovog rada nije kompletna, te za testiranje funkcionalnosti koji uključuju npr. GPS senzor ili akcelerometar treba koristiti fizički uređaj.

Iznenađujuće je da Mapsforge biblioteke omogućavaju razvoj skoro svih funkcija koje su i prisutne u Google-ovim bibliotekama, s obzirom da je ovaj projekt razvijen isključivo od strane entuzijasta i dobrovoljaca. Zaključno se može reći da Mapsforge daje programerima mogućnost da riješe bilo koji zamislivi problem u okvirima mobilne kartografije, a ukoliko je potrebna složenija navigacije ili bolji rendering, 3d rendering, može se posegnuti za dodatnim bibliotekama poput Nutiteq kako je demonstrirano u zadnjem primjeru.

[10] Sve izrađene aplikacije testirane su na uređajima LG Nexus 5, Asus Nexus 7, HTC Desire 310 i Samsung S3 GT-i8200n.

9. Literatura

1. Meier, Reto: Professional Android 4 Application Development, WROX, 2012.

2. Hardy, Brian; Phillips, Bill: Android Programming: The Big Nerd Ranch guide, Big Nerd Ranch, 2013.

3. Cadenhead, Roger; Lemay, Laura: Java 6, Naučite za 21 dan, Kompjuter biblioteke, 2007.

4. Google Maps API v2 službena dokumentacije (zadnja provjera 2.9.2014.):[https://developers.google.com/maps/documentation/android]

5. Mapsforge službena dokumentacija (zadnja provjera 2.9.2014.):

   [https://code.google.com/p/mapsforge/]

9.1. Reference

1. [Ref 1]: Steuer, Jonathan: Defining Virtual Reality: Dimensions Determining Telepresence, Department of Communication, Stanford University,1993

10. Popis slika

11. Popis priloga

[1] Digitalna verzija završnog rada u *pdf* formatu

[2] Izvorni kod (Eclipse projekt) aplikacije „Hello Map“

[3] Izvorni kod (Eclipse projekt) aplikacije „Rijeka Offline Maps“

[4] Izvorni kod (Eclipse projekt) aplikacije „Navigacija Rijeka“

[5] Potpisani izvršni kod aplikacije „Hello Map“ u *apk* formatu za instalaciju na Android uređajima

[6] Potpisani izvršni kod aplikacije „Rijeka Offline Maps“ u *apk* formatu za instalaciju na Android uređajima

[7] Potpisani izvršni kod aplikacije „Navigacija Rijeka“ u *apk* formatu za instalaciju na Android uređajima